CST126 W19 Money, Money, Money

Money.

# Introduction

In this lab we will be working with arrays of characters (c-strings) that are supposed to represent money. We’ll check to see that it really is money and do a couple of editing tasks.

# Learning objectives

* Pointers, especially passing into functions
* Pointer Arithmetic

# Requirements

* I have provided a money.h. You will implement the functions listed in money.h in the blank money.cpp.
* You must use the signatures defined in money.h.
* Note that I provide constants for array size for you. Use them. (No literals for array size)
* You may only use [] when you define arrays. Otherwise you must use pointer arithmetic.
* You may not use the c-string (ie strcpy, strlen) functions in money.cpp. Instead use pointer arithmetic.
* You can implement either remove\_all\_spaces or is\_money first. Add\_decimal should be implemented last as it calls the other two functions.
* Once you figure out the first routine, the rest of them will be easier. If you can do the pointer arithmetic one place you can transfer your knowledge (via copy and paste. 😊) to another place.
* The main program has tests that I have already created for you. You can turn on the tests for each of the three functions in money.h by updating the #define constants in main. For example, if you are ready to test the function "remove\_all\_spaces" then set TEST\_REMOVE\_SPACE to 1.
* I've provided a space where you can add your own tests. It is proceeded with a "Student added tests – Grader may ignore". This is for your own intermediate work. The grader will ignore it.

**remove\_all\_spaces**

* void remove\_all\_spaces(char \* target, const char \* source, int size);
* Removes any and all spaces from the source and places the result in target using pointer arithmetic.
* Doesn't care if the source is money or not.
* Makes sure that the '\0' is in the proper place on target.

**is\_money**

* bool is\_money(const char \* source, int size);
* Determines if the string in source is money and returns true if it is, false if it isn't.
* Money is any string that only contains the following:
  + Digits between 0-9
  + The decimal point (.)
  + $
  + Comma (,)
* You have a choice, either remove all the spaces first, OR include space as one of the things that you check.
* We aren't going to be particularly sophisticated with this check. Position of the character doesn't matter. With this check the following things would be valid:
  + 123…123
  + 12,,123.00
  + 12$12

**add\_decimal**

* bool add\_decimal(char \* target, const char \* source, int size);
* removes all the spaces from source and places it in target.
* Checks that the source is money
  + If it isn't money it skips adding the decimal. Target is the same as source except that the spaces have been removed.
  + If it is money then add\_decimal adds .00 to it if it doesn't already have a decimal. Places result in target.
* add\_decimal makes sure that the target is big enough to add the decimal.
  + If there is NOT enough space, target is the same as source except that the spaces have been removed.
* add\_decimal makes sure that the '\0' is in the proper place on target.

# Output

In the main program I have provided tests for all the functions. You only need to make your output match what's listed below. You are not allowed to change the tests. You can add your own tests below the line that says "Student added tests – Grader may ignore.". This output was produced by turning on all the define constants.

